```
title: "Exercise-10..Study and implementation of data transpose operation."
Roll# & Name: MCA_SET28_Velu,(67220200080)
date: '2022-07-07'
output: pdf_document
output:pdf_document: default
```

```
######Exercise 10############
#Aim: Study and implementation of data transpose operations in R

#Sample Data
data <- read.table(text="X Y Z
ID12 2012-06 566
ID1 2012-06 10239
ID6 2012-06 524
ID12 2012-07 2360
ID1 2012-07 13853
ID6 2012-07 2352
ID12 2012-08 3950
ID1 2012-08 14738
ID6 2012-08 4104",header=TRUE)
data
```

```
##      X       Y     Z
## 1 ID12 2012-06   566
## 2  ID1 2012-06 10239
## 3  ID6 2012-06   524
## 4 ID12 2012-07  2360
## 5  ID1 2012-07 13853
## 6  ID6 2012-07  2352
## 7 ID12 2012-08  3950
## 8  ID1 2012-08 14738
## 9  ID6 2012-08  4104
```

```
#Install reshape2 package if not installed already
if (!require(reshape2)){
  install.packages('reshape2')
  library(reshape2)
}
```

```
## Loading required package: reshape2
```

```
## Warning: package 'reshape2' was built under R version 4.2.1
```

```
#R Code : Transform Long to Wide Format
mydt = dcast(data,X~Y,value.var = "Z")
mydt
```

```
##      X 2012-06 2012-07 2012-08
## 1  ID1   10239   13853   14738
## 2 ID12     566    2360    3950
## 3  ID6     524    2352    4104
```

```
#library(reshape2)
data1 <- read.table(text="year semiyear product income
1 1 productA 13377
1 2 productA 14069
1 1 productB 11426
1 2 productB 11750
2 1 productA 11122
2 2 productA 11202
2 1 productB 14712
2 2 productB 10169",header=TRUE)
data1
```

```
##   year semiyear  product income
## 1    1        1 productA  13377
## 2    1        2 productA  14069
## 3    1        1 productB  11426
## 4    1        2 productB  11750
## 5    2        1 productA  11122
## 6    2        2 productA  11202
## 7    2        1 productB  14712
## 8    2        2 productB  10169
```

```
xx=dcast(data1, year + semiyear ~ product, value.var = "income")
xx
```

```
##   year semiyear productA productB
## 1    1        1    13377    11426
## 2    1        2    14069    11750
## 3    2        1    11122    14712
## 4    2        2    11202    10169
```

```
#If you want the final output to be reported at year level

dcast(data1, year ~ product, value.var = "income")
```

```
## Aggregation function missing: defaulting to length
```

```
##   year productA productB
## 1    1        2        2
## 2    2        2        2
```

```
#The income values are incorrect in the above table.
dcast(data1, year ~ product, fun.aggregate = sum, value.var = "income")
```

```
##   year productA productB
## 1    1    27446    23176
## 2    2    22324    24881
```

```
#Convert Wide Format Data to Long Format
#Create Sample Data
mydata = read.table(text= "ID setosa versicolor virginica
1 5.1 NA NA
2 4.9 NA NA
3 NA 7 NA
4 NA 6.4 NA
5 NA NA 6.3
6 NA NA 5.8
", header=TRUE)
mydata
```

```
##   ID setosa versicolor virginica
## 1  1    5.1         NA        NA
## 2  2    4.9         NA        NA
## 3  3     NA        7.0        NA
## 4  4     NA        6.4        NA
## 5  5     NA         NA       6.3
## 6  6     NA         NA       5.8
```

```
#The following program would reshape data from wide to long format.
library(reshape2)
x = colnames(mydata[,-1])
t = melt(mydata,id.vars = "ID",measure.vars = x , variable.name="Species",
         value.name="Sepal.Length",na.rm = TRUE)
t
```

```
##    ID    Species Sepal.Length
## 1   1     setosa          5.1
## 2   2     setosa          4.9
## 9   3 versicolor          7.0
## 10  4 versicolor          6.4
## 17  5  virginica          6.3
## 18  6  virginica          5.8
```

```
#Sample Data
data = read.table(text="
X Y Z
6 5 0
6 3 NA
6 1 5
8 5 3
1 NA 1
8 7 2
2 0 2", header=TRUE)
data
```

```
##   X  Y  Z
## 1 6  5  0
## 2 6  3 NA
## 3 6  1  5
## 4 8  5  3
```

```
## 5 1 NA  1
## 6 8  7  2
## 7 2  0  2
```

```
#Apply Function
#Calculate maximum value across row
apply(data, 1, max)
```

```
## [1]  6 NA  6  8 NA  8  2
```

```
#It returns NA if NAs exist in a row. To ignore NAs, you can use the following line of code.
apply(data, 1, max, na.rm = TRUE)
```

```
## [1] 6 6 6 8 1 8 2
```

```
#Calculate mean value across row
apply(data, 1, mean)
```

```
## [1] 3.666667       NA 4.000000 5.333333       NA 5.666667 1.333333
```

```
apply(data, 1, mean, na.rm = TRUE)
```

```
## [1] 3.666667 4.500000 4.000000 5.333333 1.000000 5.666667 1.333333
```

```
#Calculate number of 0s in each row
apply(data == 0, 1, sum, na.rm= TRUE)
```

```
## [1] 1 0 0 0 0 0 1
```

```
#Calculate number of values greater than 5 in each row
apply(data > 5, 1, sum, na.rm= TRUE)
```

```
## [1] 1 1 1 1 0 2 0
```

```
#Select all rows having mean value greater than or equal to 4
df = data[apply(data, 1, mean, na.rm = TRUE)>=4,]
df
```

```
##   X Y  Z
## 2 6 3 NA
## 3 6 1  5
## 4 8 5  3
## 6 8 7  2
```

```
#Remove rows having NAs
helper = apply(data, 1, function(x){any(is.na(x))})
df2 = data[!helper,]

df2
```

```
##   X Y Z
## 1 6 5 0
## 3 6 1 5
## 4 8 5 3
## 6 8 7 2
## 7 2 0 2
```

```
#It can be easily done with df2 = na.omit(data).
#Count unique values across row
df3 = apply(data,1, function(x) length(unique(na.omit(x))))
df3
```

```
## [1] 3 2 3 3 1 3 2
```